(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(51) 。Int. Cl. [7]
G06F 15/00

(11) 공개번호 특2002-0021237
(43) 공개일자 2002년03월20일

(21) 출원번호 10-2000-0053958
(22) 출원일자 2000년09월14일

(71) 출원인 (주)마하넷
유찬영
대전 서구 둔산2동 1207번지

(72) 발명자 유찬형
대전광역시유성구신성동한울아파트106-405
윤상원
대전광역시서구둔산동샘머리아파트102-304
송효재
대전광역시서구갈마동1170번지B04호
박현제
대전광역시서구갈마동831번지301호

(74) 대리인 특허법인 신성

심사청구 : 있음

# (54) 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를제공하는 실시간 미들웨어 장치 및 그 서비스 방법

요약

본 발명은 임베디드 시스템(Embedded System)의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어에 관한 것으로서, 원격 프로시져 호출 (RPC: Remote Procedure Calls), 명명 서비스(Naming Service), 인터페이스 정의 언어 (IDL: Interface Definition Language) 와 컴파일러 그리고, 클러스터링(Clustering) 기능 등을 제공함으로써 임베디드 시스템(Embedded System) 상에서의 소프트웨어 개발 생산성 향상과 플랫폼의 독립적이고 유연한 시뮬레이션 제공, 그리고 실시간 수행에서의 신뢰성을 향상 시키는 것을 그 특징으로 한다.

대표도
도 1

색인어

미들웨어, 인터페이스, 쓰레드, 외부데이터표현, 명명서비스

명세서

도면의 간단한 설명

도 1은 본 발명에 있어서의 실시간 미들웨어의 기능 및 인터페이스에 대하여 설명하기 위한 구성 예시도.

도 2는 본 발명에 따른 명명 서비스의 계층적인 구성 예시도.

도 3은 본 발명에 따른 실시간 미들웨어의 메시지 헤더 프로토콜 구성 예시도.

도 4는 본 발명에 따른 원격 프로시져 호출 프로토콜 구성 예시도.

도 5는 본 발명에 따른 원격 프로그램 수행 프로토콜 구성 예시도.

도 6은 본 발명에 따른 원격 메모리 접근 프로토콜 구성 예시도.

도 7은 본 발명에 따른 원격 파일 접근 프로토콜 구성 예시도.

도 8은 본 발명에 따른 원격 객체 접근 프로토콜 구성 예시도.

도 9는 본 발명에 따른 명명 서비스의 일실시예 처리 흐름도.

도 10은 본 발명에 따른 실시간 미들웨어 서버의 서비스 일실시예 처리 흐름도.

도 11은 본 발명에 따른 클라이언트의 원격 프로시져 호출 방법에 대한 일실시예 처리 흐름도.

도 12는 본 발명에 따른 서버의 원격 프로시져 호출 처리 방법에 대한 일실시예 처리 흐름도.

도 13은 본 발명에 따른 IDL 규약의 구성 예시도.

< 도면의 주요 부분에 대한 부호의 설명 >

101 : 임베디드 시스템(Embedded System)

102 : 범용 컴퓨터

110 : 서비스를 구현하기 위한 클라이언트/서버간의 인터페이스들(Interfaces)

120 : 서비스의 요청과 그 응답이 전달되는 통로

130 : 실시간 미들웨어 내부 지원 기능들 (Facilities)

201 : 주 명명 서비스 서버(Master Name Server)

202 : 하위 명명 서비스 서버(Second Group Name Server)

203 : 기능 서버

204 : 클라이언트

205 : 상위 명명 서비스 서버(First Group Name Server)

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명은 분산 요소 기반의 소프트웨어 개발 방법으로서의 미들웨어에 관한 것으로서, 특히 임베디드 시스템(Embedded System)에서의 분산 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치 및 그 서비스 방법에 관한 것이다.

일반적으로 분산 시스템은 다층의 클라이언트/서버 구조의 실질적인 구조를 취한다. 즉, 비즈니스 계층과 자료 접근 계층간의 분리가 아니라 시스템 내의 구성요소나 다른 시스템 상에 있는 구성요소에서 제공하는 어떤 서비스라도 이용 가능하도록 애플리케이션의 모든 기능을 드러내는 투명한 시스템(Transparent System)이라 할 수 있으며, 이러한 분산 시스템을 가능하도록 지원해주는 것이 미들웨어이다. 미들웨어는 분산형 컴퓨팅을 보다 쉽게 구현한다는 목표 하에 여러 기종간의 차이를 극복하도록 범용적으로 개발된 소프트웨어라고 할 수 있다.

또한, 실시간 미들웨어란 시간제약, 스케쥴링, 서비스 신뢰성, 장애에 대한 허용 등의 요소를 갖춘 미들웨어를 의미한다. 여기서, 시간제약 요소라 함은 외부의 서비스 요청에 대하여 제한 시간 내에 처리가 이루어지도록 하는 기능으로서, 만일 제한된 시간 내에 결과를 출력시키지 못하면 그 시스템은 주어진 작업을 수행하는 데 실패하였다고 말할 수 있다. 한편, 스케쥴링 요소라 함은 일정한 자원에 대한 접근을 정해진 순서에 따라 처리해 주는 것으로서, 다수의 클라이언트가 서버에게 서비스를 요청하였을 때 해당 서비스들이 시간적으로 충돌을 일으키지 않고 정확하게 수행될 수 있도록 조절하는 기능이다. 또한, 서비스 신뢰성 요소란 프로그램, 시스템 또는 하드웨어 장치가 일정시간 이상 주어진 기능을 정상적으로 수행할 수 있는 능력을 의미한다. 마지막으로 장애에 대한 허용 요소란 시스템 내의 일정 부분에 장애가 발생했을 때, 해당 장애가 시스템의 중단을 초래하지 않도록 하는 기능을 의미한다.

보통 소프트웨어 개발자들이 개발하는 과정에서 부딪히게 되는 두 가지 환경적인 문제점이 있는데, 하나는 이기종간의 네트워킹과 운영 문제이고, 다른 하나는 네트워크 애플리케이션 개발시 운영체제의 설비 부족문제이다. 즉, 두개 혹은 그 이상의 시스템 간에 프로세싱을 실행하기 위해서는 서로 다른 운영체제나 서버 소프트웨어간 이종의 네트워킹 혹은 프로토콜을 위해 필요한 내용을 전달하는 통신 기능이 필요하다. 이 때, 각기 다른 하드웨어, DBMS (Data Base Management System), 프로토콜, 애플리케이션, 데이터 포맷 등을 중간에서 효율적으로 연결하는 역할을 하는 것이 미들웨어의 기능이다.

미들웨어가 속한 종래 기술들은 다음과 같다.

RPC(Remote Procedure Calls)는 미들웨어 중에서 가장 광범위하게 사용되며, 기타 다른 미들웨어 시스템의 하부 구조로 이용되는 미들웨어이다. RPC는 사용자에게 원격지에 있는 함수를 호출할 수 있는 방법을 제공한다. RPC는 IDL(Interface Definition Language) 과 IDL 컴파일러를 제공하는데 사용자는 IDL을 이용해서 원하는 함수를 작성한다. 이러한 RPC를 제공하는 것으로는 Sun/USL ONC(Open Network Computing)에서 제공하는 RPC와 DCE(Distributed Computing Environment)에서 제공하는 RPC를 그 예로 들 수 있다.

메시지 큐잉 미들웨어 (Message Oriented Middleware, MOM) 는 비연결적인 비동기 통신을 제공한다. 즉, 클라이언트와 서버의 연결은 로컬 큐와 원격 큐 사이에서 보내기(Send)와 받기(Receive) 명령을 서로 교환함으로써 구축 되는데, 클라이언트와 서버는 메시지에 대한 응답을 기다리지 않고 계속 수행된다. 통신에 관련된 일은 큐 매니저(Queue Manager)가 관장한다.

종래의 마이크로소프트 사에서 개발한 객체(Object) 기반 미들웨어인 DCOM(Distributed Component Object Model)는 COM(Component Object Model)이 네트워크에 걸쳐 분산되어 있다. 그리고 객체 클라이언트는 마치 객체가 자신의 어드레스 영역에 있는 것처럼 객체를 이해한다. 즉, 객체가 로컬에 있든 원격 기계에 있든지 개의치 않고 다룰 수 있는 서비스를 제공한다. 그러나, 이 DCOM은 오로지 윈도우즈(Windows) 플랫폼에서만 지원된다는 한계가 있다.

또한, 자바(Java) RMI(Remote Method Invocation)는 자바 객체를 사용하는 분산 프로그래밍을 위한 간단하고 직접적인 모델을 제공한다. RMI를 기반으로 하는 시스템은 어떠한 자바 가상 머신에서도 높은 이식과 수행이 가능하다.

한편, CORBA(Common Object Request Broker Architecture) 는 OMG(Object Management Group)라는 컨소시엄에서 제안한 표준안으로서, 객체 지향 기술을 기반으로 한 분산 애플리케이션 프로그램을 통합할 수 있는 표준 기술이라 할 수 있다. CORBA에서는 컴포넌트의 인터페이스를 정의하기 위한 표준 메커니즘을 제공하며, 개발자의 특정 언어를 사용하는 이들 인터페이스의 구현을 쉽게 하는 도구들을 제공한다.

그러나, 앞에서 살펴 본 객체 지향 기술을 기반으로 한 종래의 객체 미들웨어인 DCOM, Java RMI, CORBA 등은 비록 다양한 서비스들과 높은 신뢰성을 갖고는 있다 하겠으나, 객체 지향 개발 방법에 대한 사전 이해를 반드시 필요로 하며, 현실적으로 개발자에게 많은 훈련과 노력을 요구한다는 문제점을 내포하고 있다. 아울러, 위에서 언급한 미들웨어들은 범용 컴퓨터 환경만을 주요 대상으로 하여 개발되고 적용되어 왔다. 따라서, 임베디드 시스템(Embedded System) 이 갖는 많은 제약과 특성들을 제대로 반영하지 못하는 한계를 갖고 있다. 예를 들어, 미들웨어를 수행하기 위하여 필요로 하는 많은 메모리와 CPU 등의 자원이 불충분하고, 한 이유로 인한 실시간 시스템이 요구하는 처리 성능을 제대로 발휘하지 못하고 있다. 또한, 프로시져 또는 객체의 속성(attribute)과 연산(operation)에 대한 인터페이스(Interface)만을 제공함으로 인해 파일 접근, 프로그램 수행, 쓰레드 수행 등을 위해서는 각각 별도의 솔루션을 필요로 하는 문제가 있다.

## 발명이 이루고자 하는 기술적 과제

본 발명은 상기한 바와 같은 종래의 제반 문제점들을 해결하기 위하여 제안된 것으로서, 임베디드 시스템(Embedded System)상에서 원격 프로시져 호출(RPC) 방식을 기반으로 명명 서비스(Naming Service)와 인터페이스 정의 언어(IDL) 컴파일러를 결합함으로써 실시간 성능을 보장하고, 플랫폼 및 네트워크 프로토콜 등에 대하여 독립성을 확보하며, 시스템 및 소프트웨어 요소들의 재구성을 용이하게 하는 임베디드 시스템(Embedded System)의 통합 프레임워크를 제공하는 실시간 미들웨어 장치 및 그 서비스 방법을 제공하는데 그 목적이 있다.

또한, 본 발명은 분산된 임베디드 시스템(Embedded System)의 특성이 반영된 인터페이스를 제공하며, 실시간 시스템으로서의 기능을 보장하기 위하여 시간초과(Timeout), 쓰레드(Thread), 라이브러리를 제공하고, 시스템과 소프트웨어 요소들의 유연한 재구성을 위하여 명명 서비스(Naming Service)를 통하여 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치 및 그 서비스방법을 제공하는데 또 다른 목적이 있다.

본 발명의 또 다른 목적은 실시간 미들웨어를 사용하여 모든 요소들을 구성할 경우에 별도의 장비없이 소결합 장애 허용 시스템(Loosely coupled fault tolerant system)을 실현할 수 있도록 하는 클러스터링 수단을 이용하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치 및 그 서비스방법을 제공하고자 함에 있다.

여기서, 본 발명이 제공하는 각 기능들에 대한 정의는 다음과 같다.

1.명명 서비스 : 시스템 내부에 존재하는 자원(즉, 서버)에 대하여 논리적인 이름을 부여하고, 그 이름을 통하여 물리적인 주소를 반환함으로써 외부에서의 접근이 가능하도록 하는 서비스

2.클러스터링 : 복수 대의 컴퓨터를 연결하여 하나의 시스템처럼 작동하도록 만드는 방법으로서, 일반적으로 복수 대의 서버를 서로 클러스터링하여 강력한 처리가 요구되는 작업을 수행한다. 만일 클러스터링된 서버 중 하나가 다운될 경우 운영체제는 다른 서버를 대체하여 작업을 진행하므로 사용자는 서버가 다운되었는지의 여부를 인식하지 못하고 계속 작업을 진행할 수 있다.

3.쓰레드 기능(멀티 쓰레드) : CPU가 한번에 복수개의 쓰레드를 처리하는 것으로서, 하나의 소프트웨어에서 여러 기능을 실행할 수 있다는 점에서 한번에 복수개의 소프트웨어를 실행시킬 수 있는 다중작업(multitasking)과는 다른 개념이다. 이런 작업의 예로는 문서편집기에서 문서를 팩스로 전송하는 도중에 계속해서 문서 작성을 하는 것을 들 수 있다.

4.시간 초과 : 임의의 서버에 대하여 서비스를 요청할 때 서비스를 처리해야 할 서버가 비정상적으로 작동하거나 네트워크 등에서의 문제로 인하여 서비스가 불가능한 상황에서 계속적으로 서비스를 기다리는 것이 아니라, 일정 시간 내에 결과가 오지 않을 경우 오류가 발생했음을 알리고 다른 처리를 할 수 있도록 하는 것을 말한다.

5.중복 : 일반적으로 하나의 서비스는 하나의 서버에 대하여 수행되지만, 여러 서버에 대하여 해당 서비스가 동시에 수행되도록 하는 것을 말한다.

6.록 : 공유 자원을 복수개의 프로그램이 사용하는 경우 효율적이며 데이터 무결성을 보장하기 위하여 사용되는 방법으로서, 하나의 프로그램이 해당 자원을 사용하고 있을 때 다른 프로그램으로 하여금 그 사용을 금지시키는 것을 말한다.

7.라이센스 : 제품의 공급자와 사용자간의 계약사항을 유지시켜 주기 위한 기능을 말한다.

8.디버그 : 작성된 프로그램에서 발생되는 문제를 해결하는 작업을 효율적으로 할 수 있도록 지원하는 기능을 말한다.

9.외부 데이터 표현 : 여러 기계 장치간의 차이점을 극복하기 위하여 장치 독립적인 데이터 구조를 사용하는 방법을 말한다.

## 발명의 구성 및 작용

상기의 목적을 달성하기 위하여 본 발명의 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치는 클라이언트/서버간에 서비스를 요청하고 제공하기 위하여 인터페이스 기능을 수행하기 위한 인터페이스 수단 및 시스템 내부에 존재하는 자원에 대하여 논리적인 이름을 부여하고, 그 이름을 통하여 물리적인 주소를 반환함으로써 외부에서의 접근이 가능하도록 하는 명명 서비스 기능부, CPU가 한번에 복수개의 쓰레드를 처리하는 쓰레드 기능부, 요청한 서비스에 대하여 일정 시간 내에 결과가 오지 않는 경우 오류가 발생하였음을 알리는 시간초과 기능부 및 장치 독립적인 데이터 구조를 사용하는 외부 데이터 표현 기능부를 포함하는 내부 기능 제공 수단을 포함하는 것을 특징으로 한다.

또한, 본 발명의 목적을 달성하기 위하여 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치의 상기 내부 제공 수단은 복수대의 서버를 서로 클러스터링하는 클러스터링 기능부, 여러 서버에 대하여 해당 서비스가 동시에 수행되도록 하는 기능을 하는 중복 기능부, 하나의 프로그램이 해당 자원을 사용하고 있는 경우 다른 프로그램으로 하여금 그 자원의 사용을 금지시키는 기능을 하는 록 기능부, 제품의 공급자와 사용자간의 계약사항을 유지시키는 기능을 하는 라이센스 기능부 및 작성된 프로그램에서 발생되는 문제를 해결하는 작업을 효율적으로 할 수 있도록 지원하는 기능을 하는 디버그 기능부를 더 포함하는 것을 특징으로 한다.

또한, 본 발명의 목적을 달성하기 위하여 상기 클러스터링 기능부는 실시간 미들웨어를 사용하여 모든 요소들을 구성할 경우에 별도의 장비없이 소결합 장애 허용 시스템이 실현되도록 지원하는 것을 특징으로 한다.

또한, 본 발명의 목적을 달성하기 위하여 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치의 상기 인터페이스 수단은, 미들웨어 내의 클라이언트가 서비스를 요청하고 서버가 해당 서비스를 제공하기 위하여 클라이언트는 물리적으로 분리된 프로시져를 원격으로 호출하는 원격호출수단 및 서버가 그 수행의 결과를 클라이언트에게 반환하는 결과반환수단을 포함하는 것을 특징으로 한다.

또한, 본 발명의 목적을 달성하기 위하여 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치의 상기 인터페이스 수단은 원격으로 프로시져를 호출할 수 있도록 하는 원격 프로시져 호출 기능부, 원격으로 라이브러리를 호출할 수 있도록 하는 원격 라이브러리 호출 기능부, 원격으로 쓰레드를 수행할 수 있도록 하는 원격 쓰레드 수행 기능부, 원격으로 파일을 접근할 수 있도록 하는 원격 파일 접근 기능부, 원격으로 메모리를 접근할 수 있도록 하는 원격 메모리 접근 기능부, 원격으로 프로그램을 수행할 수 잇도록 하는 원격 프로그램 수행 기능부 및 원격으로 객체를 수행할 수 있도록 하는 원격 객체수행 기능부를 포함하는 것을 특징으로 한다.

또한, 본 발명의 목적을 달성하기 위하여 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치의 상기 명명 서비스 기능부는 해당 노드에서 수행되는 서버들을 분산 관리하기 위하여 임베디드 시스템 상의 각 노드에 배치되고 논리적 구성에 따라 계층적으로 배치된 적어도 하나의 하위 명명 서비스 서버(Name Server) 및 상기 하위 명명 서비스 서버를 포함하는 전체 시스템의 구성을 관리하기 위하여 최상위에 배치된 주 명명 서비스 서버(Master Name Server)를 포함하는 것을 특징으로 한다.

또한, 본 발명의 목적을 달성하기 위하여 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치의 상기 명명 서비스 기능부는 임베디드 시스템 상에서 실시간 미들웨어 내의 서비스를 제공하는 서버의 물리적인 위치가 동적으로 변경 되어도, 사전 정의된 논리적인 주소를 사용하는 클라이언트에서는 그 소프트웨어의 변경이나 재 컴파일 등의 작업 없이 서비스를 구현할 수 있도록 지원하는 것을 특징으로 한다.

또한, 본 발명의 목적을 달성하기 위하여 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치는 인터페이스 정의 언어를 컴파일하기 위한 인터페이스 정의 언어 컴파일러 수단 및 상기 인터페이스 정의 언어 컴파일러 수단에 의하여 컴파일 옵션을 플랫폼에 맞게 지정함으로써 상기 플랫폼에 대한 맵핑을 구현하기 위한 인터페이스 정의 언어(IDL) 규약 제공 수단을 더 포함하는 것을 특징으로 한다.

또한, 본 발명의 목적을 달성하기 위하여 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치의 상기 인터페이스 정의 언어 규약은 원격 서버명들(1305), 라이브러리들(1306), 상수 정의들(1307), 데이터 유형 정의들(1308), 프로시져 또는 라이브러리 정의들(1309), 메모리 정의들(1310), 쓰레드 정의들(1311) 및 프로그램 정의들(1312)을 제공하는 것을 특징으로 한다.

또한, 본 발명의 목적을 달성하기 위하여 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치의 상기 컴파일러 수단은, 소프트웨어를 개발하고 수행하고자 하는 임베디드 시스템(Embedded System)의 플랫폼과 네트워크 프로토콜 및 개발 언어 등에 관계없이 공통의 단일한 인터페이스를 정의하고 그 인터페이스를 구현함을 특징으로 한다.

또한, 본 발명의 목적을 달성하기 위하여 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치의 임베디드 시스템(Embedded System)의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어를

구현하기 위한 프로토콜은, 모든 상호 접속 수단에 공통이면서 클라이언트/서버 사이에 서비스를 요청하고 제공하기 위한 메시지 프로토콜; 원격으로 프로시져를 호출하기 위한 프로시져 프로토콜; 원격으로 라이브러리를 호출하기 위한 라이브러리 프로토콜; 원격으로 쓰레드를 호출하기 위한 프로시져 프로토콜; 원격으로 파일을 접근하기 위한 파일 프로토콜; 원격으로 메모리에 접근하기 위한 메모리 프로토콜; 원격으로 프로그램을 수행하기 위한 프로그램 프로토콜; 및 원격으로 객체를 수행하기 위한 객체 수행 프로토콜을 포함하는 것을 특징으로 한다.

또한, 본 발명의 목적을 달성하기 위하여 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 미들웨어 서비스 방법은 명명 서비스를 수행하는 명명 서비스 처리과정; 미들웨어가 실시간으로 처리를 행하는 미들웨어 서버 실시간 처리과정; 원격 프로시져 호출을 요청하기 위하여 클라이언트측에서 처리되는 클라이언트측 처리과정; 및 클라이언트가 요청한 원격 프로시져 호출 서비스의 처리를 위하여 서버측의 스케쥴링된 쓰레드에서 처리되는 서버측 처리과정을 포함하는 것을 특징으로 한다.

또한, 본 발명의 목적을 달성하기 위하여 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 미들웨어 서비스 방법의 상기 명명 서비스 처리과정은 현재의 명명 서비스 서버를 인식하여 통신을 초기화하는 제1단계; 메시지를 수신하고, 수신된 메시지에 나타나는 서비스를 처리하는 제2단계; 및 서비스에 대한 처리 결과를 전송함으로써 서비스의 수행을 완료하는 제3단계를 포함하는 것을 특징으로 한다.

또한, 본 발명의 목적을 달성하기 위하여 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 미들웨어 서비스 방법의 상기 미들웨어 서버 실시간 처리과정은 명명 서비스 서버에 등록하여 서버를 초기화하는 제1단계; 메시지를 수신하고, 수신된 메시지를 처리하는 제2단계; 처리 큐에 존재하는 서비스를 처리 가능한 쓰레드에게 스케쥴링하는 제3단계; 분할된 메시지를 조합하여 서비스를 처리하는 제4단계; 및 메시지처리이외의 주기적으로 발생하는 서비스를 처리하는 제5단계를 포함하는 것을 특징으로 한다.

또한, 본 발명의 목적을 달성하기 위하여 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 미들웨어 서비스 방법의 상기 클라이언트측 처리과정은 초기화 및 데이터를 마샬링하는 제1단계; 해당 서비스를 제공할 서버의 위치 정보를 얻어 메시지를 전송하는 제2단계; 및 모든 메시지의 전송이 완료되면 서버로부터 처리 결과값을 수신하고 처리 결과값을 반환하여 프로시져 호출을 완료하는 제3단계로 이루어진 것을 특징으로 한다.

또한, 본 발명의 목적을 달성하기 위하여 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 미들웨어 서비스 방법의 서버측 처리과정은 프로시져 수행 요청을 접수하여 해당 프로시져 ID를 검색하는 제1단계; 및 요청된 프로시져를 호출하여 처리하는 제2단계를 포함하는 것을 특징으로 한다.

이하, 첨부된 도면을 참조하여 본 발명의 일 실시 예에 따른 임베디스 시스템(Embedded System)의 통합 소프트웨어 개발 프레임워크'를 설명하면 다음과 같다.

도 1은 본 발명에 있어서의 실시간 미들웨어의 기능 및 인터페이스에 대하여 설명하기 위한 구성 예시도로서, 도면을 참조하면, 실시간 미들웨어는 명명(Naming, 131), 클러스터링(Clustering, 132), 멀티 쓰레딩(Multi-threading, 133), 록킹(Locking, 136), 시간초과(Timeout, 134), 라이센싱(Licensing, 137), 디버그(Debug, 138), 외부 데이터 표현(XDR ; eXternal Data Representation, 139), 중복(Replication, 135) 등의 내부 기능 수단(130)을 제공하며, 클라이언트(204)/기능서버(203)간에 서비스를 요청하고 제공하는 원격 프로시져 호출(Procedure, 111), 원격 라이브러리 호출(Library : 112), 원격 쓰레드 수행(Thread : 113), 원격 파일 접근(File : 114), 원격 메모리 접근(Memory : 115), 원격 프로그램 수행(Program : 116), 객체(Object) 수행(117) 기능 등을 인터페이스하는 수단을 제공함을 알 수 있다. 즉, 소프트웨어 개발시 클라이언트(204) 측에서는 실시간 미들웨어가 제공하는 상기의 인터페이스(110)를 사용하여 기능서버(203)측에 존재하는 요소들에 접근함으로써 서비스를 제공받을 수 있게 된다.

이러한 서비스들은 도1에서와 같이 범용 컴퓨터(102)에서 뿐만 아니라 임베디드 시스템(Embedded System, 101)에서도 동일한 방식으로 제공될 수 있으며, 서비스 요청을 전달하고 그에 대한 응답을 반환하기 위한 방법으로서 UDP/IP 그리고, IPC 프로토콜(120)을 사용할 수 있다. 또한, 각 서비스들은 한 시스템에서 뿐만 아니라 여러 시스템 상에 분산되어질 수 있으며, 이러한 분산 환경에서 효율적으로 서비스를 제공하기 위하여 명명 서비스(Naming Service, 131) 수단이 제공된다.

도 2는 본 발명에 따른 명명 서비스(Naming Service, 131)의 계층적인 구성 예시도로서, 주 명명 서비스 서버(Master Name Server, 201)를 정점으로 상위 및 하위 명명 서비스 서버(first group and second group Name Server)들이 트리 구조로 연결되어 있음을 나타낸다. 시스템 별로 존재하는 각각의 하위 명명 서비스 서버(202)들은 기능서버(203)의 인증 및 등록, 클라이언트의 요청에 따른 서버 주소의 제공 그리고 서버의 상태 체크(809) 등의 기능을 수행한다. 하위 명명 서비스 서버(202)들은 자신이 속한 시스템 내의 기능서버(203)들을 관리하며, 상위 명명 서비스 서버(205)들은 역시 자신이 속한 시스템 내의 서버들과 함께 자신보다 한 단계 낮은 하위 명명 서비스 서버(202)및 그에 속한 기능서버(203)들을 함께 관리한다. 최상위의 주 명명 서비스 서버(201)는 전체 시스템 내의 모든 서버에 대한 정보와 상위 명명 서비스 서버, 하위 명명 서비스 서버 및 기능서버를 포함하는 명명 서비스 서버(Name Server)들의 정보를 갖게 되며, 필요에 따라 정보를 제공할 수 있다.

도 2를 참조하여 기능서버(203)의 등록 및 서비스 제공과정을 설명하면 다음과 같다.

도 2에서 기능서버(203)가 작동된 후 인증을 요청하고 자신을 정식 등록하기 위하여, 자신이 속한 시스템 내의 하위 명명 서비스 서버(202)에게 등록을 요청한다. 주 명명 서비스 서버(201)만이 해당 기능서버(203)에 대한 인증을 처리할 수 있으므로 해당 기능서버(203)의 등록 요청은 계층적으로 상위로 전달되어 최종적으로 주 명명 서비스 서버(201)에 도달하게 된다. 주 명명 서비스 서버(201)는 인증 처리를 한 후 그 결과를 통보하게 되는데, 등록 요청이 전달된 경로를 역으로 거슬러 가서 해당 기능서버(203)에 이르게 된다. 이렇게 등록된 기능서버(203)는 비로소 자신이 갖고 있는 서비스를 제공할 수 있게 된다. 이러한 서비스를 이용하기 위하여 클라이언트(204)는 서비스를 제공할 기능서버(203)의 위치를 사전에 파악하여야 하는데, 이것은 클라이언트가 속한 하위 명명 서비스 서버에게 요구함으로써 런-타임 시에 동적 방법으로 제공받을 수 있게 된다. 여기서, 동적방법이란 다음의 의미를 내포하고 있다. 클라이언트가 특정 서버에 대하여 서비스를 요청할 때, 클라이언트는 서버의 주소를 알아야만 그 주소를 통하여 서비스를 요청하고 그 결과를 받게 되는데, 이러한 주소가 프로그램이 수행되기 전에 결정되는 경우를 정적 방법이라 하고, 프로그램이 수행되어 필요로 할 때 주소 정보를 얻게 되는 것을 동적 방법이라 한다. 이렇게 제공받은 주소 정보를 이용하여 클라이언트(204)는 해당 기능서버(203)가 제공하는 서비스를 이용할 수 있게 된다.

도 3은 본 발명에 따른 실시간 미들웨어의 메시지 헤더 프로토콜 구성 예시도로서, 클라이언트(204)와 기능서버(203) 그리고 명명 서비스 서버들(Name Server ; 201, 202)간에 송수신되는 메시지의 공통 헤더 프로토콜을 보이고 있다. 하나의 서비스는 일정한 크기의 여러 메시지로 단편화될 수 있으며, 또한 서비스는 각 서비스의 주체(201, 202, 203, 204)간에 메시지 단위로 송수신된다. 이렇게 단편화되어 수신되는 메시지는 최종 수신측에서 조립된다. 메시지 헤더는 서비스의 유형, 사용되는 네트워크 프로토콜, 서비스의 처리 상태, 서비스 처리 번호, 서비스 길이 등 서비스에 대한 기본 정보와 함께 메시지 개수, 메시지 번호, 메시지 길이 등분된 메시지의 신뢰성 있는 전송을 위한 메시지 정보들로 구성된다. 또한, 실제 서비스의 내용은 메시지 헤더 이후의 데이터 영역에 실리게 된다. 각 서비스들을 구현하기 위해서는 이러한 메시지 헤더와 함께 데이터 영역에 실리는 서비스별 세부 프로토콜을 필요로 하는데 이에 관하여는 도 4, 도 5, 도 6, 도 7, 도 8에 각각 예시되어있다.

도 4는 본 발명에 따른 원격 프로시져 호출(111) 프로토콜에 대한 구성 예시도이다.

클라이언트(204)는 호출하고자 하는 프로시져 ID와 프로시져에 전달될 인수의 개수, 각 인수들의 데이터 유형과 크기 그리고, 인수의 값 등의 정보를 기능서버(203)에 전달함으로써 원격으로 프로시져를 호출할 수 있다. 종속 서버(203)는 그러한 정보를 바탕으로 해당 프로시져를 검색하여 호출한 뒤, 그 결과값을 반환하게 된다. 프로시져의 처리 흐름도는 도 11과 도 12에 나타나 있다.

한편, 여기서 도시하지 않은 원격 라이브러리 호출(112)과 원격 쓰레드 수행(113) 서비스의 프로토콜 또한 도 4에 도시된 원격 프로시져 호출의 프로토콜 구성과 동일한 구조를 갖도록 구성할 수 있다.

도 5는 본 발명에 따른 원격 프로그램 수행(116) 프로토콜에 대한 구성 예시도로서, 클라이언트는 수행할 프로그램명과 인수들을 서버에 전달함으로써 원격으로 프로그램을 수행할 수 있게 한다.

도 6은 본 발명에 따른 원격 메모리 접근(115) 프로토콜에 대한 구성 예시도로서, 클라이언트는 접근할 메모리의 ID와 접근 유형, 옵셋 그리고 데이터를 서버에 전달함으로써 원격으로 메모리에 대해 접근할 수 있게 된다.

도 7은 본 발명에 따른 원격 파일 접근(114) 프로토콜에 대한 구성 예시도로서, 클라이언트는 접근할 파일의 디스크립터와 접근 유형 그리고, 접근 유형별 기타 정보와 데이터를 서버에 전달함으로써 원격으로 파일을 접근할 수 있게 된다.

도 8은 본 발명에 따른 원격 객체 수행(117) 프로토콜에 대한 구성 예시도로서, 클라이언트는 수행할 객체(Object)의 ID(identification)와 멤버 ID, 접근 유형, 전달될 인수의 개수, 각 인수들의 데이터 유형과 크기 그리고, 인수의 값 등의 정보를 서버에 전달함으로써 원격으로 객체(Object)를 수행할 수 있게 된다.

도 9는 본 발명에 따른 명명 서비스(Naming Service)의 일 실시예 처리 흐름도로서, 이에 대하여 구체적으로 설명하면 다음과 같다.

명명 서비스 서버는 그 위치에 따라 주 명명서비스 서버(201), 상위 명명 서비스 서버(205) 혹은 하위 명명 서비스 서버(202)로 구분된다. 주 명명 서비스 서버(201)는 디스크(903)가 존재하는 사전 지정된 시스템에서 수행되는 명명 서비스 서버로서, 디스크(903) 상에 기록되어 있는 이름표(Name Table)와 클러스터 표(Cluster Table)라는 정보를 읽어 들여 내부적으로 초기화하게 된다(902). 모든 명명 서비스 서버는 메시지의 송· 수신을 위하여 통신을 초기화 한 뒤(904), 클라이언트(204), 종속 서버(203) 또는 다른 명명 서비스 서버들로부터 메시지 수신을 대기한다(905). 메시지가 수신되면(905), 메시지가 나타내는 서비스의 종류를 판별하여(906) 서버 등록, 서버 주소 요청 혹은 서버 상태 체크 등의 서비스를 처리하고(907), 각 서비스에 대한 처리 결과를 메시지 송신측에 전송한다(908). 명명 서비스 서버 자신의 상태를 출력할 주기가 되면(909), 서버 주소 정보 리스트를 화면에 출력하게 된다(910). 하나의 서비스를 수행하면 다시 메시지를 수신하기(905) 위하여 회귀하여 위의 과정을 반복하게 된다.

도 10은 본 발명에 따른 실시간 미들웨어 서버의 서비스 일실시예 처리 흐름도로서, 상세히 설명하면 다음과 같다.

종속 서버(203)에서 통신을 위한 핸들러를 생성하고(1001), 자신이 속한 시스템의 명명 서비스 서버에게 등록을 요청한다(1002). 정상적으로 등록되었다는 결과를 수신한(1003) 이후에 클러스터링(Clustering)된 다른 서버와 메모리 동기화(1004) 및 INIT 프로시져와 쓰레드 수행(1005) 그리고 쓰레드 풀(Pool) 생성(1006) 등 초기화 작업을 수행한다. 초기화 작업이 완료된 이후에야 서비스가 가능한 상태로 되며, 클라이언트(204)로부터 메시지를 수신할 수 있게 된다(1007). 수신한 메시지가 분할된 서비스의 첫 메시지인 경우(1008), 처리 큐에 해당 서비스를 삽입한다(1009). 그리고, 수신된 메시지는 별도의 메시지 리스트에 삽입한다(1010). 수신한 메시지에 대한 처리 이후에 처리 큐를 검색하여 수행 가능한 서비스가 존재할 경우(1011), 쓰레드 풀(Pool) 내의 처리 가능한 쓰레드에게 서비스를 스케쥴링한다(1012). 스케쥴링된 쓰레드는 해당 서비스의 분할된 메시지들을 조합하고(1013), 서비스의 종류에 따라(1015) 프로시져 호출, 프로그램 수행, 메모리 접근, 파일 접근, 쓰레드 수행 등을 처리하게 된다(1016). 세부적인 서비스의 처

리와 관련해서는 대표적으로 프로시져 호출에 대해 도12에 나타나 있다(1016). 서비스 처리를 마친 후, 시간초과(Timeout)된 서비스를 삭제하고(1017) 서버 록을 점검하며(1018) 기능서버(203)에게 자신의 상태를 보고(1019) 한다. 그 이후에는 계속적으로 서비스를 수행하기 위하여 메시지 수신(1007) 작업부터 위의 과정을 반복한다.

도 11은 본 발명에 따른 클라이언트의 원격 프로시져 호출 방법에 대한 일실시예 처리 흐름도로서, 이를 구체적으로 설명하면 다음과 같다.

서비스를 요청하는 클라이언트(204)에서 핸들러를 생성한 뒤(1101), 해당 프로시져의 스터브(Stub)를 호출하면(1102) 스터브(Stub) 내부에서 호출하고자 하는 프로시져의 ID가 부여된다(1103). 프로시져 ID는 개발자에 의해 명시적으로 부여되거나, IDL 컴파일러에 의해 자동으로 부여되는 프로시져 고유 번호이다. 그리고, 서비스 데이터를 구성하는 프로시져 호출시 전달되는 인수들을 마샬링하여 하나의 서비스 데이터를 작성한다(1104). 서비스는 그 서비스의 총 길이가 최대 허용 길이를 초과하지 않는 범위 내에서 실현될 수 있다(1105). 해당 서비스를 제공할 서버의 위치 정보를 얻기 위하여 명명 서비스 서버(Name Server)에게 요청하고(1106) 서비스의 각 메시지들을 순서대로 전송한다(1107). 데이터의 신뢰성을 위하여 각 메시지가 전송된 뒤 서버는 해당 메시지의 정상 수신 여부를 통보하며 클라이언트는 이 ACK를 수신한(1108) 이후에야 다음 메시지를 전송하게 된다(1109). 서버가 모든 메시지를 처리한 후 클라이언트가 서버로부터 처리 결과값 및 시간 초과를 수신하여(1110) 시간을 초과하여도 결과값이 수신되지 않은 지의 여부를 판단하고(1111), 시간 내에 수신되면 결과값을 반환함으로써 프로시져 호출을 완료한다(1112). 만일 지정된 오류 발생 시간을 초과하여도 결과값이 수신되지 않는 경우에는 오류값을 반환하게 된다(1113).

도 12는 본 발명에 따른 서버의 원격 프로시져 호출 처리 방법에 대한 일실시예 처리 흐름도로서, 클라이언트(204)가 요청한 원격 프로시져 호출 서비스를 처리하는 기능 서버측의 스케쥴링된 쓰레드에서 처리되는 과정에 대한 것이며, 이를 구체적으로 설명하면 다음과 같다.

프로시져 수행 요청을 접수하면(1201), 서비스 정보에 실려 있는 프로시져 ID와(1202) 프로시져 인수를 얻는다(1203). 프로시져 테이블을 검색하여(1204) 해당 프로시져 ID가 존재하는지 검사한(1205) 뒤, 요청된 프로시져를 호출하고(1206) 그 반환값을 마샬링하여(1207) 클라이언트에 그 결과를 전송하게 된다(1208). 만일 해당 프로시져 ID가 존재하지 않는 경우는 오류값을 전송한다(1209).

도 13는 본 발명에 따른 IDL 규약의 구성 예시도이다.

IDL 정의는 파일로 저장되며, 각 파일당 하나의 서버명을 갖는다(1301). 그 서버명은 기능서버(203)의 실제 이름이 된다. 지정한 서버가 가질 수 있는 서비스들을 서버명으로 시작하는 블록 내에 정의할 수 있는 바, 서버의 유형이 노말(NORMAL), 라이브러리(LIBRARY) 또는 파일(FILE)인지를 지정할 수 있으며(1302), 서버의 유형이 LIBRARY인 경우 라이브러리 서버의 텍스트 주소(Text Address)와 라이브러리 서버의 테이블 주소(Table Address)를 지정할 수 있다(1303, 1304). 그리고, 참조하는 원격 서버명들과 참조하는 라이브러리들을 지정함으로써 소프트웨어 개발을 용이하게 할 수 있도록 지원한다(1305, 1306). 아울러, 상수 정의들, 데이터 유형 정의들, 프로시져 또는 라이브러리 정의들, 메모리 정의들, 쓰레드 정의들 그리고, 프로그램 정의들을 지정할 수 있다.(1307, 1308, 1309, 1310, 1311, 1312)

이상에서 설명한 본 발명은, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에 있어 본 발명의 기술적 사상을 벗어나지 않는 범위 내에서 여러 가지 치환, 변형 및 변경이 가능하므로 전술한 실시예 및 첨부된 도면에 한정되는 것이 아니다.

## 발명의 효과

본 발명의 임베디드 시스템(Embedded System) 소프트웨어 통합 개발 프레임워크로서의 실시간 미들웨어는, 임베디드 시스템(Embedded System) 소프트웨어 개발에 있어 플랫폼과 네트워크 프로토콜 그리고 개발 언어에 대한 독립적인 환경을 제공함으로써 개발 시간을 크게 단축해 주며, 새로운 기술의 신속한 도입을 가능케 하여 그 경쟁력을 높이는 효과가 있다.

또한, 각 플랫폼과 네트워크 프로토콜 등에 대해서 심도 있는 이해를 위한 내/외부 교육 및 시스템 프로그램 작성 등을 위하여 별도의 시간 및 경비 투여나 노력 없이 사전 개발되고 검증된 실시간 미들웨어를 사용함으로써 그 기능을 쉽게 적용할 수 있도록 지원해 준다.

아울러, 범용 컴퓨터 환경에 비해 상대적으로 취약한 개발 환경을 갖고 있는 임베디드 시스템(Embedded System) 상에서 개발되고 수행 되어야 할 소프트웨어를 범용 컴퓨터 상에서 보다 쉽게 시뮬레이션을 할 수 있도록 한 이 기능을 통해 범용 컴퓨터 상에서 지원되는 많은 개발 도구를 사용할 수 있도록 해 주며, 소프트웨어가 수행 되어야 하나 임베디드 시스템(Embedded System)이 갖춰 있지 않은 공간 또는 환경에서도 중단 없는 개발을 가능케 해 준다. 아울러 여러 대의 장비가 연동되어야 하는 시스템의 개발에 있어서도 단 한 대의 장비로도 그것을 가능케 해 준다.

특히, 명명 서비스(Naming Service)를 이용함으로써 시스템의 구성(Architecture)을 자유롭게 재구성 (restructuring) 하는 것이 용이하다. 초기에 설계한 시스템의 구성에 따라 개발된 소프트웨어는 설계된 시스템의 구성 중 많은 부분이 종속적이다. 이러한 종속성은 시험 및 실제 수행을 통해 발생하는 문제의 해결을 위하여 또는 평가의 결과로 합리적인 시스템 구성을 새롭게 반영하기 위하여 시스템을 재구성 하고자 하는 경우, 많은 부분이 새롭게 개발 되거나 코드의 수정이 불가피해지게 되며 그러한 변경의 정당성을 검증하기 위하여 시험을 다시 수행 해야만 하는데, 이러한 상황에서 명명 서비스(Naming Service)는 시스템을 구성하는 소프트웨어 요소들의 위치와 상태 등을 정적.동적으로 관리해 줌으로써 새로운 개발이나 코드의 수정 없이도 재구성을 효과적으로 반영할 수 있게 해 주는 매우 우수한 효과를 갖는다.

(57) 청구의 범위

청구항 1.

클라이언트/서버간에 서비스를 요청하고 제공하기 위하여 인터페이스 기능을 수행하기 위한 인터페이스 수단; 및

시스템 내부에 존재하는 자원에 대하여 논리적인 이름을 부여하고, 그 이름을 통하여 물리적인 주소를 반환함으로써 외부에서의 접근이 가능하도록 하는 명명 서비스 기능부, CPU가 한번에 복수개의 쓰레드를 처리하는 쓰레드 기능부, 요청한 서비스에 대하여 일정 시간 내에 결과가 오지 않는 경우 오류가 발생하였음을 알리는 시간초과 기능부 및 장치 독립적인 데이터 구조를 사용하는 외부 데이터 표현 기능부를 포함하는 내부 기능 제공 수단

을 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치.

청구항 2.

제1항에 있어서,

상기 내부 기능 제공 수단은 복수대의 서버를 서로 클러스터링하는 클러스터링 기능부

를 더 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치

청구항 3.

제2항에 있어서,

상기 클러스터링 기능부는

실시간 미들웨어를 사용하여 모든 요소들을 구성할 경우에 별도의 장비없이 소결합 장애 허용 시스템이 실현되도록 지원하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치.

청구항 4.

제2항에 있어서,

상기 내부 기능 제공 수단은 여러 서버에 대하여 해당 서비스가 동시에 수행되도록 하는 기능을 하는 중복 기능부

를 더 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치

청구항 5.

제4항에 있어서,

상기 내부 기능 제공 수단은 하나의 프로그램이 해당 자원을 사용하고 있는 경우 다른 프로그램으로 하여금 그 자원의 사용을 금지시키는 기능을 하는 록 기능부

를 더 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치

청구항 6.

제1항에 있어서,

상기 내부 기능 제공 수단은 여러 서버에 대하여 해당 서비스가 동시에 수행되도록 하는 기능을 하는 중복 기능부

를 더 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치

청구항 7.

제5항에 있어서,

상기 내부 기능 제공 수단은 하나의 프로그램이 해당 자원을 사용하고 있는 경우 다른 프로그램으로 하여금 그 자원의 사용을 금지시키는 기능을 하는 록 기능부

를 더 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치

청구항 8.

제1항에 있어서,

상기 내부 기능 제공 수단은 하나의 프로그램이 해당 자원을 사용하고 있는 경우 다른 프로그램으로 하여금 그 자원의 사용을 금지시키는 기능을 하는 록 기능부

를 더 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치

청구항 9.

제1항 내지 제8항 중 어느 한 항에 있어서,

상기 내부 기능 제공 수단은 제품의 공급자와 사용자간의 계약사항을 유지시키는 기능을 하는 라이센스 기능부; 및

작성된 프로그램에서 발생되는 문제를 해결하는 작업을 효율적으로 할 수 있도록 지원하는 기능을 하는 디버그 기능부

를 더 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치

청구항 10.

제9항에 있어서,

상기 인터페이스 수단은, 미들웨어 내의 클라이언트가 서비스를 요청하고 서버가 해당 서비스를 제공하기 위하여 클라이언트는 물리적으로 분리된 프로시저를 원격으로 호출하는 원격호출수단; 및

서버가 그 수행의 결과를 클라이언트에게 반환하는 결과반환수단

을 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치.

청구항 11.

제9항에 있어서, 상기 인터페이스 수단은,

원격으로 프로시저를 호출할 수 있도록 하는 원격 프로시저 호출 기능부;

원격으로 라이브러리를 호출할 수 있도록 하는 원격 라이브러리 호출 기능부;

원격으로 쓰레드를 수행할 수 있도록 하는 원격 쓰레드 수행 기능부;

원격으로 파일을 접근할 수 있도록 하는 원격 파일 접근 기능부;

원격으로 메모리를 접근할 수 있도록 하는 원격 메모리 접근 기능부;

원격으로 프로그램을 수행할 수 잇도록 하는 원격 프로그램 수행 기능부; 및

원격으로 객체를 수행할 수 있도록 하는 원격 객체수행 기능부

를 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치.

청구항 12.

제 11항에 있어서,

상기 명명 서비스 기능부는,

해당 노드에서 수행되는 서버들을 분산 관리하기 위하여 임베디드 시스템 상의 각 노드에 배치되고 논리적 구성에 따라 계층적으로 배치된 적어도 하나의 하위 명명 서비스 서버(Name Server); 및

상기 하위 명명 서비스 서버를 포함하는 전체 시스템의 구성을 관리하기 위하여 최상위에 배치된 주 명명 서비스 서버(Master Name Server)

를 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치.

청구항 13.

제 11항에 있어서,

상기 명명 서비스 기능부는,

임베디드 시스템 상에서 실시간 미들웨어 내의 서비스를 제공하는 서버의 물리적인 위치가 동적으로 변경되어도, 사전 정의된 논리적인 주소를 사용하는 클라이언트에서는 그 소프트웨어의 변경이나 재 컴파일 등의 작업 없이 서비스를 구현할 수 있도록 지원하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치.

청구항 14.

제12항에 있어서,

인터페이스 정의 언어를 컴파일하기 위한 인터페이스 정의 언어 컴파일러 수단; 및

상기 인터페이스 정의 언어 컴파일러 수단에 의하여 컴파일 옵션을 플랫폼에 맞게 지정함으로써 상기 플랫폼에 대한 맵핑을 구현하기 위한 인터페이스 정의 언어(IDL) 규약 제공 수단

을 더 포함하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치.

청구항 15.

제14항에 있어서,

상기 컴파일러 수단은, 소프트웨어를 개발하고 수행하고자 하는 임베디드 시스템(Embedded System)의 플랫폼과 네트워크 프로토콜 및 개발 언어 등에 관계없이 공통의 단일한 인터페이스를 정의하고 그 인터페이스를 구현함을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치.

청구항 16.

제14항에 있어서, 상기 자체적인 인터페이스 정의 언어 규약 제공 수단은,

원격 서버명들(1305), 라이브러리들(1306), 상수 정의들(1307), 데이터 유형 정의들(1308), 프로시저 또는 라이브러리 정의들(1309), 메모리 정의들(1310), 쓰레드 정의들(1311) 및 프로그램 정의들(1312)을 제공하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 장치.

청구항 17.

임베디드 시스템(Embedded System)의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어를 구현하기 위한 프로토콜은,

모든 상호 접속 수단에 공통이면서 클라이언트/서버 사이에 서비스를 요청하고 제공하기 위한 메시지 프로토콜;

원격으로 프로시저를 호출하기 위한 프로시저 프로토콜;

원격으로 라이브러리를 호출하기 위한 라이브러리 프로토콜;

원격으로 쓰레드를 호출하기 위한 프로시저 프로토콜;

원격으로 파일을 접근하기 위한 파일 프로토콜;

원격으로 메모리에 접근하기 위한 메모리 프로토콜;

원격으로 프로그램을 수행하기 위한 프로그램 프로토콜; 및

원격으로 객체를 수행하기 위한 객체 수행 프로토콜

을 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 미들웨어 서비스 방법.

청구항 18.

실시간 미들웨어 시스템에서 적용하기 위한

명명 서비스를 수행하는 명명 서비스 처리과정;

미들웨어가 실시간으로 처리를 행하는 미들웨어 서버 실시간 처리과정;

원격 프로시저 호출을 요청하기 위하여 클라이언트측에서 처리되는 클라이언트측 처리과정; 및

클라이언트가 요청한 원격 프로시저 호출 서비스의 처리를 위하여 서버측의 스케쥴링된 쓰레드에서 처리되는 서버측 처리과정

을 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 미들웨어 서비스 방법

청구항 19.

제18항에 있어서, 상기 명명 서비스 처리과정은,

현재의 명명 서비스 서버를 인식하여 통신을 초기화하는 제1단계;

메시지를 수신하고, 수신된 메시지에 나타나는 서비스를 처리하는 제2단계; 및

서비스에 대한 처리 결과를 전송함으로써 서비스의 수행을 완료하는 제3단계

를 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 서비스 방법.

청구항 20.

제 19항에 있어서, 제1단계는,

현재의 명명 서비스 서버가 주 명명 서비스 서버(Master Name Server, 201)인지의 여부를 판단하는 단계(901);

상기 주 명명 서비스 서버(Master Name Server, 201)가 디스크(903) 상의 정보를 읽어 들여 내부적으로 초기화(902)하는 단계(902); 및

메시지의 송수신을 위하여 모든 명명 서비스 서버(Name Server)가 통신을 초기화(904)하는 단계(904)

를 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 서비스 방법.

청구항 21.

제19항에 있어서, 제2단계는,

클라이언트(204), 서버(203) 또는 다른 명명 서비스 서버(Name Server)로부터 메시지를 수신하는 단계(905);

수신된 메시지가 나타내는 서비스의 종류를 판별하는 단계(906); 및

서비스의 종류에 따라 서버 등록서비스, 서버 주소 요청 서비스, 서버 상태 체크 서비스 중 하나를 처리하는 단계(907)

를 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 서비스 방법.

청구항 22.

제19항에 있어서, 제3단계는,

각 서비스에 대한 처리 결과를 메시지 송신측에 전송하는 단계(908);

명명 서비스 서버(Name Server)가 자신의 상태를 출력할 주기가 되었는지를 판단하는 단계(909);

서버 주소 정보 리스트를 화면에 출력하는 단계(910); 및

서버 등록, 서버 주소 요청, 서버 상태 체크 중 하나의 서비스의 수행을 완료(908 혹은 909)한 후에는 다시 메시지를 수신하는 단계(905)로 회귀하는 단계

를 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 실시간 미들웨어 서비스 방법.

청구항 23.

제18항에 있어서, 미들웨어 서버 실시간 처리과정은,

명명 서비스 서버에 등록하여 서버를 초기화하는 제1단계;

메시지를 수신하고, 수신된 메시지를 처리하는 제2단계;

처리 큐에 존재하는 서비스를 처리 가능한 쓰레드에게 스케쥴링하는 제3단계;

분할된 메시지를 조합하여 서비스를 처리하는 제4단계; 및

메시지처리이외의 주기적으로 발생하는 서비스를 처리하는 제5단계

를 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 미들웨어 서비스 방법

청구항 24.

제23항에 있어서, 제1단계는,

서버(203)가 작동되면 통신을 위한 핸들러를 생성하는 단계(1001);

자신이 속한 시스템의 명명 서비스 서버(Name Server)에게 등록을 요청하는 단계(1002);

정상적으로 등록되었다는 결과를 수신하는 단계(1003);

클러스터링(Clustering)된 다른 서버와 메모리 동기화를 이루는 단계(1004);

초기화를 위하여 INIT 프로시져와 쓰레드를 수행하는 단계(1005); 및

초기화를 위하여 쓰레드 Pool을 생성하는 단계(1006)

를 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 미들웨어 서비스 방법

청구항 25.

제23항에 있어서, 제2단계는,

클라이언트(204)로부터 메시지를 수신하는 단계(1007);

수신한 메시지가 분할된 서비스의 첫 메시지인지의 여부를 판단하는 단계(1008);

첫 메시지이면 처리 큐에 해당 서비스를 삽입하고, 아니면 삽입하지 아니하는 단계(1009); 및

수신된 메시지를 별도의 메시지 리스트에 삽입하는 단계(1010)

를 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 미들웨어 서비스 방법

청구항 26.

제23항에 있어서, 제3단계는,

수신한 메시지에 대한 처리 이후에 처리 큐를 검색하여 수행가능한 서비스가 존재하는지의 여부를 판단하여 수행가능한 서비스가 없으면 메시지를 수신하는 단계(1007)로 회귀하는 단계(1011); 및

수행가능한 서비스가 있으면 쓰레드 Pool 내의 처리 가능한 쓰레드에게 서비스를 스케쥴링하는 단계(1012)

를 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 미들웨어 서비스 방법

청구항 27.

제23항에 있어서, 제4단계는,

스케쥴링된 쓰레드가 해당 서비스의 분할된 메시지들을 조합하는 단계(1013);

메시지 조합의 오류 여부를 판단하여 오류가 있으면 메시지를 수신하는 단계(1007)로 복귀하는 단계(1014);

메시지 조합에 오류가 없으면 서비스의 종류를 판단하는 단계(1015); 및

서비스의 종류에 따라 서비스를 처리하는 단계(1016)

를 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 미들웨어 서비스 방법

청구항 28.

제23항에 있어서, 제5단계는,

서비스 처리를 완료한 후 시간초과(Timeout)된 서비스를 삭제하는 단계(1017);

서버 록을 점검하는 단계(1018);

명명 서비스 서버(Name Server, 203)에게 자신의 상태를 보고하는 단계(1019); 및

이후에는 계속적인 서비스의 수행을 위하여 메시지를 수신하는 단계(1007)로 회귀하는 단계

를 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 미들웨어 서비스 방법

청구항 29.

제 23항 내지 제 27항 중 어느 한 항에 있어서,

상기 수신된 메시지는 프로시져 호출, 프로그램 수행, 메모리 접근, 파일 접근, 쓰레드 수행 중 하나

임을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 미들웨어 서비스 방법

청구항 30.

제18항에 있어서, 상기 클라이언트측 처리과정은,

초기화 및 데이터를 마샬링하는 제1단계;

해당 서비스를 제공할 서버의 위치 정보를 얻어 메시지를 전송하는 제2단계; 및

모든 메시지의 전송이 완료되면 서버로부터 처리 결과값을 수신하고 처리 결과값을 반환하여 프로시져 호출을 완료하는 제3단계

를 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 미들웨어 서비스 방법.

청구항 31.

제30항에 있어서, 제1단계는,

서비스를 요청하는 클라이언트(204)에서 핸들러를 생성하는 단계(1101);

해당 프로시져의 스터브(Stub)를 호출하는 단계(1102);

스터브(Stub) 내부에서 호출하고자 하는 프로시져의 ID가 부여되는 단계(1103);

프로시져 호출시 전달되는 인수들을 마샬링하여 하나의 서비스 데이터를 작성하는 단계(1104); 및

작성된 서비스 데이터의 총 길이가 최대 허용 길이를 초과하는 지를 판단하는 단계(1105)

를 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 미들웨어 서비스 방법

청구항 32.

제30항에 있어서, 제2단계는,

초과하지 않으면 해당 서비스를 제공할 서버의 위치 정보를 얻는 단계(1106);

서버로 서비스의 각 메시지들을 순서대로 전송하는 단계(1107);

각 메시지가 전송된 뒤 서버가 해당 메시지의 정상 수신 여부를 통보하며 클라이언트는 이 ACK를 수신한 후 다음 메시지를 전송하는 단계(1108); 및

클라이언트가 모든 메시지를 전송하였는지를 판단하여 아니면 서버로 메시지를 전송하는 단계(1107)로 복귀하는 단계(1109)

를 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 미들웨어 서비스 방법

청구항 33.

제30항에 있어서, 제3단계는,

서버가 모든 메시지를 처리한 후 클라이언트가 서버로부터 처리 결과값 및 시간 초과를 수신하는 단계(1110);

시간을 초과하여도 결과값이 수신되지 않은지의 여부를 판단하는 단계(1111);

시간 내에 수신되면 결과값을 반환하여 프로시져 호출을 완료하는 단계(1112); 및

지정된 오류 발생 시간을 초과하여도 결과값이 수신되지 않는 경우에는 오류값을 반환하는 단계(1113)

를 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 미들웨어 서비스 방법

청구항 34.

제18항에 있어서, 상기 서버측 처리과정은,

프로시져 수행 요청을 접수하여 해당 프로시져 ID를 검색하는 제1단계; 및

요청된 프로시져를 호출하여 처리하는 제2단계

를 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 미들웨어 서비스 방법

청구항 35.

제34항에 있어서, 제1단계는,

프로시져 수행 요청을 접수하는 단계(1201);

서비스 정보에 실려 있는 프로시져 ID를 얻는 단계(1202);

프로시져 인수를 얻는 단계(1203);

프로시져 테이블을 검색하는 단계(1204); 및

해당 프로시져 ID가 존재하는지 검사하는 단계(1205)

를 포함하는 것을 특징으로 하는 임베디드 시스템의 통합 소프트웨어 개발 프레임워크를 제공하는 미들웨어 서비스 방법

청구항 36.

제34항에 있어서, 제2단계는,

존재하면 요청된 프로시져를 호출하는 단계(1206);

반환값을 마샬링하는 단계(1207);

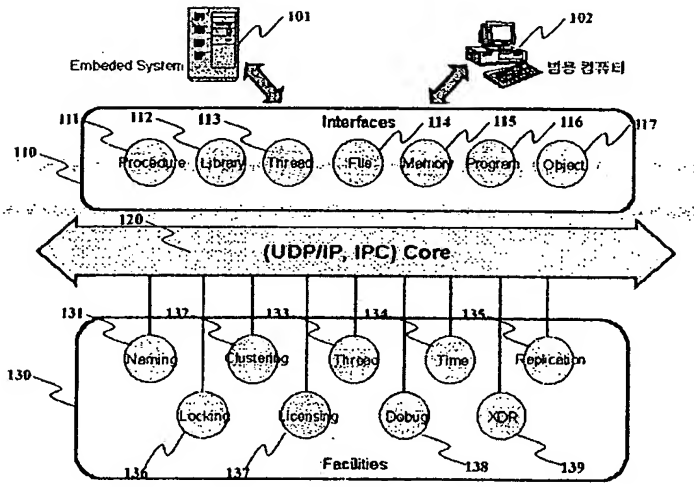
클라이언트에 그 결과를 전송하는 단계(1208); 및

만일 해당 프로시저 ID가 존재하지 않는 경우는 오류값을 전송하는 단계(1209)

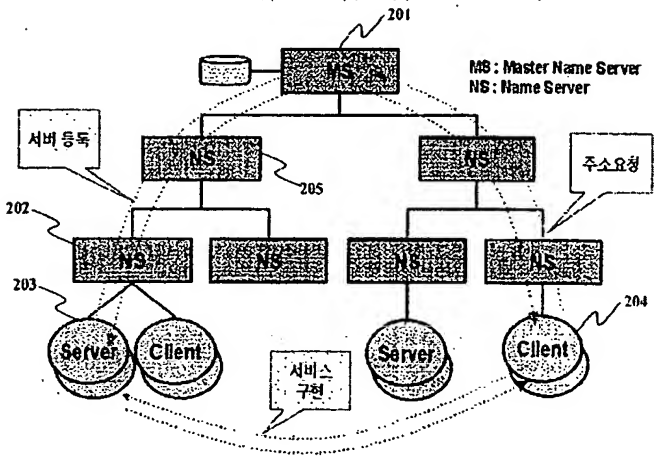
를 포함하는 것을 특징으로 하는 임베디드 시스템(Embedded System)의 통합 소프트웨어 개발 프레임워크를 제공하는 미들웨어 서비스 방법.

도면

도면 1

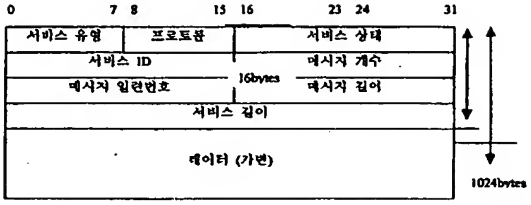



도면 2



도면 3

• 메시지 헤더 프로토콜

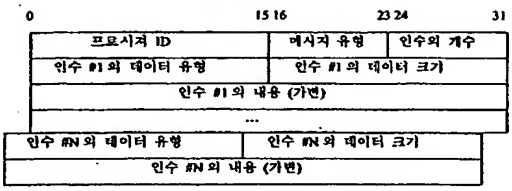



• 각 항목의 내용

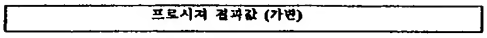
| 항 목 | 내 용 |
|---|---|
| 서비스 유형 | 서버에 요청하는 서비스의 유형을 나타냄.<br>프로시저, 메모리, 프로그램, 파일, 쓰레드 |
| 프로토콜 | UDP/IP, IPC |
| 서비스 상태 | 서비스의 수행 상태를 기록. |
| 서비스 ID | 서버로부터 할당된 서비스 처리 번호 |
| 메시지 개수 | 일정 크기로 단편화된 메시지의 개수 |
| 메시지 일련번호 | 단편화 되어 송.수신 되는 메시지의 일련번호 |
| 메시지 길이 | 해당 메시지의 길이 |
| 서비스 길이 | 서비스의 전체 길이 |

도면 4

• 프로시저 호출 요청 프로토콜




• 프로시저 호출 응답 프로토콜

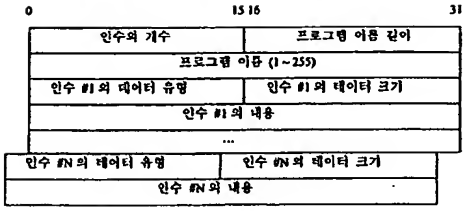



• 각 항목의 내용

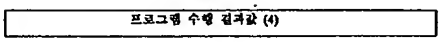
| 항 목 | 내 용 |
|---|---|
| 프로시저 ID | 수행을 요청하는 프로시저의 고유 번호 |
| 메시지 유형 | 인수 또는 반환 여부를 나타냄 |
| 인수의 개수 | 해당 프로시저 인수의 개수 |
| 데이터 유형 | 각 인수의 데이터 유형 |
| 데이터 크기 | 각 인수의 데이터 크기 |
| 인수의 내용 | 각 인수의 값 |
| 프로시저 결과값 | 프로시저 호출의 결과값 |

## 도면 5

• 프로그램 수행 요청 프로토콜




• 프로그램 수행 응답 프로토콜

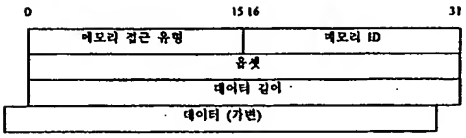



• 각 항목의 내용

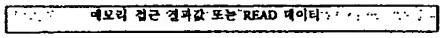
| 항 목 | 내 용 |
|---|---|
| 인수의 개수 | 해당 프로시져 인수의 개수 |
| 프로그램 이름 길이 | 수행할 프로그램의 경로명 길이 |
| 프로그램 이름 | 수행할 프로그램의 경로명 |
| 데이터 유형 | 각 인수의 데이터 유형 |
| 데이터 크기 | 각 인수의 데이터 크기 |
| 인수의 내용 | 각 인수의 값 |
| 프로그램 수행 결과 | 프로그램 수행의 결과값 |

도면 6

• 메모리 접근 요청 프로토콜




• 메모리 접근 응답 프로토콜

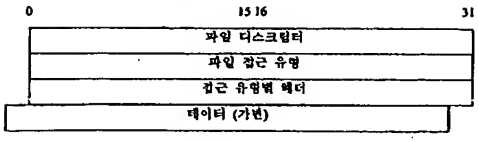



• 각 항목의 내용

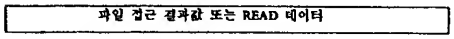
| 항 목 | 내 용 |
|---|---|
| 메모리 접근 유형 | READ/WRITE |
| 메모리 ID | 접근할 메모리의 고유 번호 |
| 옵셋 | 메모리의 시작 주소로부터의 옵셋 |
| 데이터 길이 | READ/WRITE 길이 |
| 데이터 | WRITE 데이터 |
| 결과값 | WRITE : 메모리 접근 결과값 (4)<br>READ : 데이터 (가변) |

도면 7

• 파일 접근 요청 프로토콜




• 파일 접근 응답 프로토콜

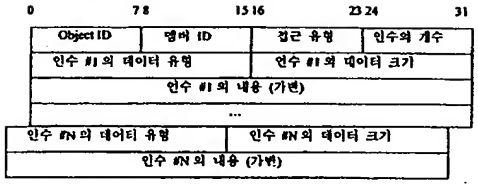



• 각 항목의 내용

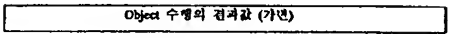
| 항 목 | 내 용 |
|---|---|
| 파일 디스크립터 | 파일 접근을 위한 식별자 |
| 파일 접근 유형 | OPEN / READ / WRITE / SEEK / LOCK / CLOSE |
| 접근 유형별 헤더 | 각 파일 접근 유형별 헤더 |
| 데이터 | 각 파일 접근 유형별 데이터 |
| 결과값 | READ: 데이터 (가변)<br>기타 : 파일 접근 결과값 (4) |

## 도면 8

* Object 수행 요청 프로토콜



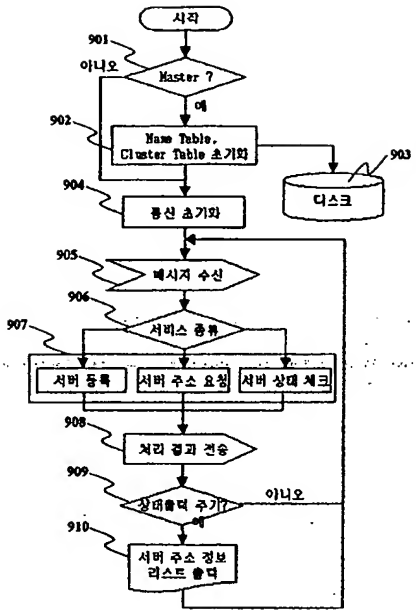

* Object 수행 응답 프로토콜



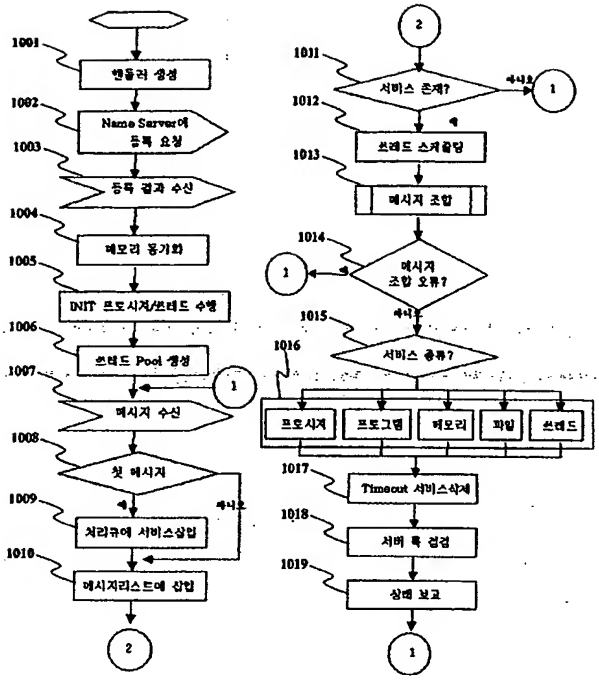

* 각 항목의 내용

| 항 목 | 내 용 |
|---|---|
| Object ID | 수행을 요청하는 Object 의 고유 번호 |
| 멤버 ID | Object 의 attribute, operation 의 고유 번호 |
| 접근 유형 | attribute read/write, operation 접근 여부를 나타냄 |
| 인수의 개수 | 인수의 개수 |
| 데이터 유형 | 각 인수의 데이터 유형 |
| 데이터 크기 | 각 인수의 데이터 크기 |
| 인수의 내용 | 각 인수의 값 |
| Object 수행 결과값 | Object 수행의 결과값 |

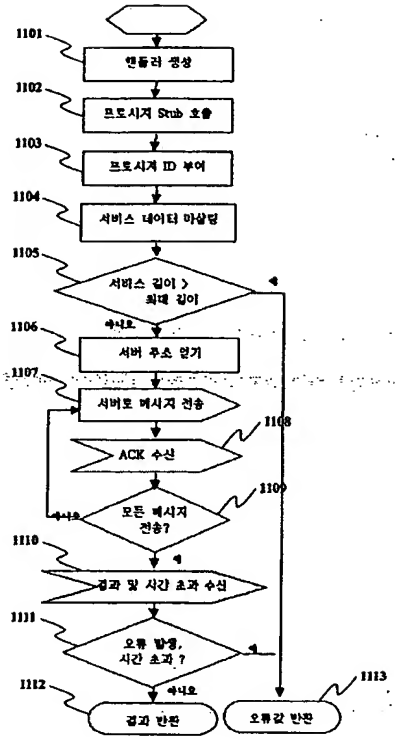
도면 9


시작
901
Master ?
아니오
예
902
Name Table,
Cluster Table 초기화
903
디스크
904
통신 초기화
905
메시지 수신
906
서비스 종류
907
서버 등록
서버 주소 요청
서버 상태 체크
908
처리 결과 전송
909
상태출력 주기?
아니오
예
910
서버 주소 정보
리스트 출력

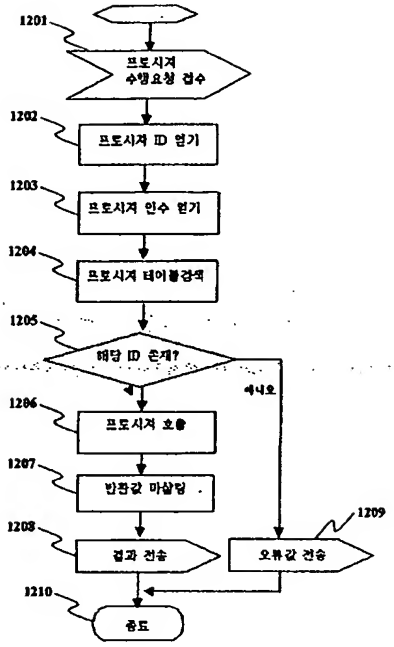
도면 10


1001
핸들러 생성
1002
Name Server에 등록 요청
1003
등록 결과 수신
1004
메모리 동기화
1005
INIT 프로시저/쓰레드 수행
1006
쓰레드 Pool 생성
1007
메시지 수신
1008
첫 메시지
1009
처리큐에 서비스삽입
1010
메시지리스트에 삽입
1011
서비스 존재?
1012
쓰레드 스케쥴링
1013
메시지 조합
1014
메시지 조합 오류?
1015
서비스 종류?
1016
프로시저
프로그램
메모리
파일
쓰레드
1017
Timeout 서비스삭제
1018
서버 룩 검검
1019
상태 보고
예
아니오

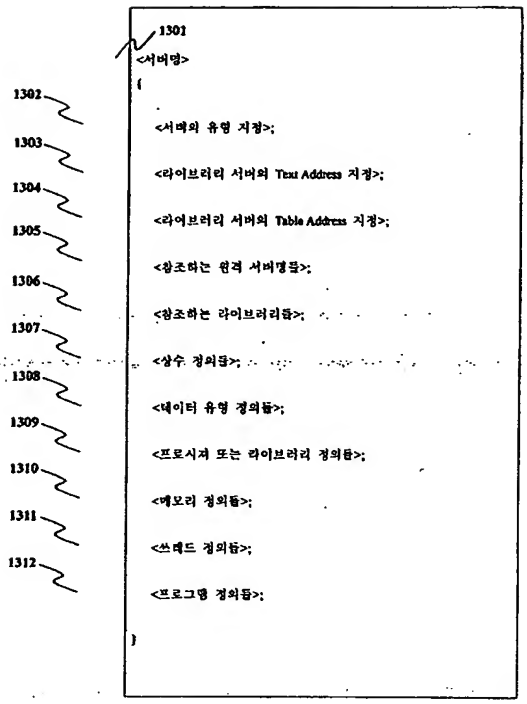
도면 11


1101
핸들러 생성
1102
프로시저 Stub 호출
1103
프로시저 ID 부여
1104
서비스 데이터 마샬링
1105
서비스 길이 > 최대 길이
예
아니오
1106
서버 주소 얻기
1107
서버로 메시지 전송
1108
ACK 수신
1109
모든 메시지 전송?
아니오
예
1110
결과 및 시간 초과 수신
1111
오류 발생, 시간 초과 ?
예
아니오
1112
결과 반환
1113
오류값 반환

도면 12


1201
프로시저
수행요청 접수
1202
프로시저 ID 읽기
1203
프로시저 인수 읽기
1204
프로시저 테이블검색
1205
해당 ID 존재?
아니오
1206
프로시저 호출
1207
반환값 마샬링
1208
결과 전송
1209
오류값 전송
1210
종료

도면 13


1301
<서버명>
{
1302
<서버의 유형 지정>;
1303
<라이브러리 서버의 Text Address 지정>;
1304
<라이브러리 서버의 Table Address 지정>;
1305
<참조하는 원격 서버명들>;
1306
<참조하는 라이브러리들>;
1307
<상수 정의들>;
1308
<데이터 유형 정의들>;
1309
<프로시져 또는 라이브러리 정의들>;
1310
<메모리 정의들>;
1311
<쓰레드 정의들>;
1312
<프로그램 정의들>;
}